Package 'word2vec'

June 12, 2020

Type Package

Title Distributed Representations of Words

Version 0.2.1

Maintainer Jan Wijffels <jwijffels@bnosac.be>

Description Learn vector representations of words by continuous bag of words and skip-gram implementations of the 'word2vec' algorithm.
The techniques are detailed in the paper ``Distributed Representations of Words and Phrases and their Compositionality" by Mikolov et al. (2013), available at <arXiv:1310.4546>.

URL https://github.com/bnosac/word2vec

License Apache License (>= 2.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Depends R (>= 2.10)

Imports Rcpp (>= 0.11.5)

LinkingTo Rcpp, RcppProgress

Suggests udpipe

NeedsCompilation yes

Author Jan Wijffels [aut, cre, cph] (R wrapper),
BNOSAC [cph] (R wrapper),
Max Fomichev [ctb, cph] (Code in src/word2vec)

Repository CRAN

Date/Publication 2020-06-12 09:10:02 UTC

R topics documented:

`as.matrix.word2vec` *Get the word vectors of a word2vec model*

Description

Get the word vectors of a word2vec model as a dense matrix.

Usage

```
## S3 method for class 'word2vec'
as.matrix(x, encoding = "UTF-8", ...)
```

Arguments

`x`	a word2vec model as returned by `word2vec` or `read.word2vec`
`encoding`	set the encoding of the row names to the specified encoding. Defaults to 'UTF-8'.
`...`	not used

Value

a matrix with the word vectors where the rownames are the words from the model vocabulary

See Also

`word2vec`, `read.word2vec`

Examples

```
path  <- system.file(package = "word2vec", "models", "example.bin")
model <- read.word2vec(path)

embedding <- as.matrix(model)
```

predict.word2vec *Predict functionalities for a word2vec model*

Description

Get either

- the embedding of words
- the nearest words which are similar to either a word or a word vector

Usage

```
## S3 method for class 'word2vec'
predict(
  object,
  newdata,
  type = c("nearest", "embedding"),
  top_n = 10L,
  encoding = "UTF-8",
  ...
)
```

Arguments

object	a word2vec model as returned by word2vec or read.word2vec
newdata	for type 'embedding', newdata should be a character vector of words for type 'nearest', newdata should be a character vector of words or a matrix in the embedding space
type	either 'embedding' or 'nearest'. Defaults to 'nearest'.
top_n	show only the top n nearest neighbours. Defaults to 10.
encoding	set the encoding of the text elements to the specified encoding. Defaults to 'UTF-8'.
...	not used

Value

depending on the type, you get a different result back:

- for type nearest: a list of data.frames with columns term, similarity and rank indicating with words which are closest to the provided newdata words or word vectors. If newdata is just one vector instead of a matrix, it returns a data.frame
- for type embedding: a matrix of word vectors of the words provided in newdata

See Also

word2vec, read.word2vec

Examples

```
path  <- system.file(package = "word2vec", "models", "example.bin")
model <- read.word2vec(path)
emb <- predict(model, c("bus", "toilet", "unknownword"), type = "embedding")
emb
nn  <- predict(model, c("bus", "toilet"), type = "nearest", top_n = 5)
nn

# Do some calculations with the vectors and find similar terms to these
emb <- as.matrix(model)
vector <- emb["buurt", ] - emb["rustige", ] + emb["restaurants", ]
predict(model, vector, type = "nearest", top_n = 10)

vector <- emb["gastvrouw", ] - emb["gastvrij", ]
predict(model, vector, type = "nearest", top_n = 5)

vectors <- emb[c("gastheer", "gastvrouw"), ]
vectors <- rbind(vectors, avg = colMeans(vectors))
predict(model, vectors, type = "nearest", top_n = 10)
```

`read.word2vec` *Read a binary word2vec model from disk*

Description

Read a binary word2vec model from disk

Usage

```
read.word2vec(file, normalize = FALSE)
```

Arguments

`file`	the path to the model file
`normalize`	logical indicating to normalize the embeddings by dividing by the factor (sqrt(sum(x . x) / length(x))). Defaults to FALSE.

Value

an object of class w2v which is a list with elements

- model: a Rcpp pointer to the model
- model_path: the path to the model on disk
- dim: the dimension of the embedding matrix
- n: the number of words in the vocabulary

Examples

```
path  <- system.file(package = "word2vec", "models", "example.bin")
model <- read.word2vec(path)
vocab <- summary(model, type = "vocabulary")
emb <- predict(model, c("bus", "naar", "unknownword"), type = "embedding")
emb
nn  <- predict(model, c("bus", "toilet"), type = "nearest")
nn

# Do some calculations with the vectors and find similar terms to these
emb <- as.matrix(model)
vector <- emb["gastvrouw", ] - emb["gastvrij", ]
predict(model, vector, type = "nearest", top_n = 5)
vectors <- emb[c("gastheer", "gastvrouw"), ]
vectors <- rbind(vectors, avg = colMeans(vectors))
predict(model, vectors, type = "nearest", top_n = 10)
```

`read.wordvectors` *Read word vectors from a word2vec model from disk*

Description

Read word vectors from a word2vec model from disk into a dense matrix

Usage

```
read.wordvectors(
  file,
  type = c("bin", "txt"),
  n = .Machine$integer.max,
  normalize = FALSE,
  encoding = "UTF-8"
)
```

Arguments

`file`	the path to the model file
`type`	either 'bin' or 'txt' indicating the `file` is a binary file or a text file
`n`	integer, indicating to limit the number of words to read in. Defaults to reading all words.
`normalize`	logical indicating to normalize the embeddings by dividing by the factor (sqrt(sum(x . x) / length(x))). Defaults to FALSE.
`encoding`	encoding to be assumed for the words. Defaults to 'UTF-8'

Value

A matrix with the embeddings of the words. The rownames of the matrix are the words which are by default set to UTF-8 encoding.

Examples

```
path  <- system.file(package = "word2vec", "models", "example.bin")
embed <- read.wordvectors(path, type = "bin", n = 10)
embed <- read.wordvectors(path, type = "bin", n = 10, normalize = TRUE)
embed <- read.wordvectors(path, type = "bin")

path  <- system.file(package = "word2vec", "models", "example.txt")
embed <- read.wordvectors(path, type = "txt", n = 10)
embed <- read.wordvectors(path, type = "txt", n = 10, normalize = TRUE)
embed <- read.wordvectors(path, type = "txt")
```

`word2vec` *Train a word2vec model on text*

Description

Construct a word2vec model on text. The algorithm is explained at https://arxiv.org/pdf/1310.4546.pdf

Usage

```
word2vec(
  x,
  type = c("cbow", "skip-gram"),
  dim = 50,
  window = ifelse(type == "cbow", 5L, 10L),
  iter = 5L,
  lr = 0.05,
  hs = FALSE,
  negative = 5L,
  sample = 0.001,
  min_count = 5L,
  split = c(" \n,.-!?:;/\"#$%&'()*+<=>@[]\^_`{|}~\t\v\f\r", ".\n?!"),
  stopwords = character(),
  threads = 1L,
  ...
)
```

Arguments

`x`	a character vector with text or the path to the file on disk containing training data
`type`	the type of algorithm to use, either 'cbow' or 'skip-gram'. Defaults to 'cbow'
`dim`	dimension of the word vectors. Defaults to 50.
`window`	skip length between words. Defaults to 5.
`iter`	number of training iterations. Defaults to 5.
`lr`	initial learning rate also known as alpha. Defaults to 0.05

hs	logical indicating to use hierarchical softmax instead of negative sampling. Defaults to FALSE indicating to do negative sampling.
negative	integer with the number of negative samples. Only used in case hs is set to FALSE
sample	threshold for occurrence of words. Defaults to 0.001
min_count	integer indicating the number of time a word should occur to be considered as part of the training vocabulary. Defaults to 5.
split	a character vector of length 2 where the first element indicates how to split words and the second element indicates how to split sentences in x
stopwords	a character vector of stopwords to exclude from training
threads	number of CPU threads to use. Defaults to 1.
...	further arguments passed on to the C++ function w2v_train - for expert use only

Details

Some advice on the optimal set of parameters to use for training as defined by Mikolov et al.

- argument type: skip-gram (slower, better for infrequent words) vs cbow (fast)
- argument hs: the training algorithm: hierarchical softmax (better for infrequent words) vs negative sampling (better for frequent words, better with low dimensional vectors)
- argument dim: dimensionality of the word vectors: usually more is better, but not always
- argument window: for skip-gram usually around 10, for cbow around 5
- argument sample: sub-sampling of frequent words: can improve both accuracy and speed for large data sets (useful values are in range 0.001 to 0.00001)

Value

an object of class w2v_trained which is a list with elements

- model: a Rcpp pointer to the model
- data: a list with elements file: the training data used, stopwords: the character vector of stopwords, n
- vocabulary: the number of words in the vocabulary
- success: logical indicating if training succeeded
- error_log: the error log in case training failed
- control: as list of the training arguments used, namely min_count, dim, window, iter, lr, skip-gram, hs, negative, sample, split_words, split_sents, expTableSize and expValueMax

References

https://github.com/maxoodf/word2vec, https://arxiv.org/pdf/1310.4546.pdf

See Also

predict.word2vec, as.matrix.word2vec

Examples

```
library(udpipe)
## Take data and standardise it a bit
data(brussels_reviews, package = "udpipe")
x <- subset(brussels_reviews, language == "nl")
x <- tolower(x$feedback)

## Build the model get word embeddings and nearest neighbours
model <- word2vec(x = x, dim = 15, iter = 20)
emb   <- as.matrix(model)
head(emb)
emb <- predict(model, c("bus", "toilet", "unknownword"), type = "embedding")
emb
nn  <- predict(model, c("bus", "toilet"), type = "nearest", top_n = 5)
nn

## Get vocabulary
vocab <- summary(model, type = "vocabulary")

# Do some calculations with the vectors and find similar terms to these
emb <- as.matrix(model)
vector <- emb["buurt", ] - emb["rustige", ] + emb["restaurants", ]
predict(model, vector, type = "nearest", top_n = 10)

vector <- emb["gastvrouw", ] - emb["gastvrij", ]
predict(model, vector, type = "nearest", top_n = 5)

vectors <- emb[c("gastheer", "gastvrouw"), ]
vectors <- rbind(vectors, avg = colMeans(vectors))
predict(model, vectors, type = "nearest", top_n = 10)

## Save the model to hard disk
path <- "mymodel.bin"

write.word2vec(model, file = path)
model <- read.word2vec(path)




##
## Example getting word embeddings
##   which are different depending on the parts of speech tag
## Look to the help of the udpipe R package
##   to get parts of speech tags on text
##
library(udpipe)
data(brussels_reviews_anno, package = "udpipe")
x <- subset(brussels_reviews_anno, language == "fr")
x <- subset(x, grepl(xpos, pattern = paste(LETTERS, collapse = "|")))
x$text <- sprintf("%s/%s", x$lemma, x$xpos)
x <- subset(x, !is.na(lemma))
```

```
x <- paste.data.frame(x, term = "text", group = "doc_id", collapse = " ")
x <- x$text

model <- word2vec(x = x, dim = 15, iter = 20, split = c(" ", ".\n?!"))
emb   <- as.matrix(model)
nn    <- predict(model, c("cuisine/NN", "rencontrer/VB"), type = "nearest")
nn
nn    <- predict(model, c("accueillir/VBN", "accueillir/VBG"), type = "nearest")
nn
```

`word2vec_similarity` *Similarity between word vectors as used in word2vec*

Description

The similarity between word vectors is defined as the square root of the average inner product of the vector elements (sqrt(sum(x . y) / ncol(x))) capped to zero

Usage

```
word2vec_similarity(x, y, top_n = +Inf)
```

Arguments

`x`	a matrix with embeddings where the rownames of the matrix provide the label of the term
`y`	a matrix with embeddings where the rownames of the matrix provide the label of the term
`top_n`	integer indicating to return only the top n most similar terms from y for each row of x. If `top_n` is supplied, a data.frame will be returned with only the highest similarities between x and y instead of all pairwise similarities

Value

By default, the function returns a similarity matrix between the rows of `x` and the rows of `y`. The similarity between row i of `x` and row j of `y` is found in cell `[i, j]` of the returned similarity matrix. If `top_n` is provided, the return value is a data.frame with columns term1, term2, similarity and rank indicating the similarity between the provided terms in `x` and `y` ordered from high to low similarity and keeping only the top_n most similar records.

See Also

`word2vec`

Examples

```
x <- matrix(rnorm(6), nrow = 2, ncol = 3)
rownames(x) <- c("word1", "word2")
y <- matrix(rnorm(15), nrow = 5, ncol = 3)
rownames(y) <- c("term1", "term2", "term3", "term4", "term5")

word2vec_similarity(x, y)
word2vec_similarity(x, y, top_n = 1)
word2vec_similarity(x, y, top_n = 2)
word2vec_similarity(x, y, top_n = +Inf)

## Example with a word2vec model
path  <- system.file(package = "word2vec", "models", "example.bin")
model <- read.word2vec(path)
emb <- as.matrix(model)

x <- emb[c("gastheer", "gastvrouw", "kamer"), ]
y <- emb
word2vec_similarity(x, x)
word2vec_similarity(x, y, top_n = 3)
predict(model, x, type = "nearest", top_n = 3)
```

---

write.word2vec *Save a word2vec model to disk*

Description

Save a word2vec model as a binary file to disk or as a text file

Usage

```
write.word2vec(x, file, type = c("bin", "txt"), encoding = "UTF-8")
```

Arguments

x	an object of class w2v or w2v_trained as returned by word2vec
file	the path to the file where to store the model
type	either 'bin' or 'txt' to write respectively the file as binary or as a text file. Defaults to 'bin'.
encoding	encoding to use when writing a file with type 'txt' to disk. Defaults to 'UTF-8'

Value

a logical indicating if the save process succeeded

See Also

word2vec

Examples

```
path  <- system.file(package = "word2vec", "models", "example.bin")
model <- read.word2vec(path)


## Save the model to hard disk as a binary file
path <- "mymodel.bin"

write.word2vec(model, file = path)

## Save the model to hard disk as a text file (uses package udpipe)
library(udpipe)
path <- "mymodel.txt"

write.word2vec(model, file = path, type = "txt")
```

Index